**F14\_Chp7\_Data types and sizes\_for memory organization and data mining: Chp 6, 7 & 8**

**Deitel Chp 7 pp 283 - 310 Subbarao Wunnava 10 02 2014 Modified 10 05 2019**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

int main()

{

 printf("\n\n F14\_Chp7\_Data types and sizes\_for memory organization and data mining \n\a");

 printf(" Deitel Chp 7 pp 283 - 300 Subbarao Wunnava 10 02 2014 2880 Software \n");

 char c; // character

 short s; // short integer

 int i; // integer

 long l; // long integer

 long long ll; // long long integer

 float f; // floating decimal

 double d; // double floating decimal

 long double ld; // long double floating decimal

 int array1 [20]; // integer array1 of 20 elements

 int \*ptr1 = array1; // pointer to integer array1

 double array2 [80]; // double floating array2 of 80 elements

 int \*ptr2 = array2; // pointer to integer array2

 puts (" sizes are in bytes following 2012 Software Standards \a\n");

 printf (" \n\t\a sizeof c = %u\t sizeof(char) = %u", sizeof c, sizeof (char));

 printf (" \n\t\a sizeof s = %u\t sizeof(short) = %u", sizeof s, sizeof (short));

 printf (" \n\t\a sizeof i = %u\t sizeof(int) = %u", sizeof i, sizeof (int));

 printf (" \n\t\a sizeof l = %u\t sizeof(long) = %u", sizeof l, sizeof (long));

 printf (" \n\t\a sizeof ll = %u\t sizeof(long long) = %u", sizeof ll, sizeof (long long));

 printf (" \n\t\a sizeof f = %u\t sizeof(float) = %u", sizeof f, sizeof (float));

 printf (" \n\t\a sizeof d = %u\t sizeof(double) = %u", sizeof d, sizeof (double));

 printf (" \n\t\a sizeof ld = %u\t sizeof(long double) = %u \n", sizeof ld, sizeof (long double));

 printf (" \n\n\t\a sizeof array1 = %u\t sizeof ptr1 = %u \n", sizeof array1, sizeof ptr1);

 printf (" \n\t\a sizeof array2 = %u\t sizeof ptr2 = %u \n\n", sizeof array2, sizeof ptr2);

 return 0;

}// end main()



**EEL 2880 Fall 2016: Deitel/Subbarao: Chp 6,7,8**

**Strings, lower, upper case and addresses pointers and arrays: 10 09 2019**

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

int main()

{

 printf( " \a\n EEL 2880 Fall 2014: 10 02 2014 Deitel/Subbarao: Chp 6,7,8 \n");

 puts( "\n Strings, lower, upper case and addresses pointers and arrays: \n");

 int a [6] = {123, 234, 345, 456, 567, 678}; // 6 element array

 unsigned \*aPtr; // Pointer to 'a' array

 char s[ 80 ]; // up to 80 element character array

 unsigned \*sPtr; // Pointer to 's' array

 size\_t k, i, j; //array pointers

 aPtr = & a[0];

 sPtr = & s[0];

 printf (" \n\a decimal and hex address of 'a' array: %u, %p, \n\t \t\t and aPtr: %u, %p \n",&a, &a, aPtr, aPtr);

 printf (" \n\a decimal and hex address of 's' array: %u, %p, \n\t \t\t and sPtr: %u, %p \n",&s, &s, sPtr, sPtr);

 puts( "\n\t\a printing array [a]: \n\t");

 for (k =0; k <= 5; ++k)

 {

 printf (" \a\t %d ", a[k]);

 }

 printf (" \n\n\a decimal and hex address of end of 'a' array: %u, %p \n\t ", &a[k], &a[k]);

 puts( "\n Enter a line of text: \a\n" );

 fgets( s, 80, stdin );// obtain string of characters

 puts( "\n\a convert to upper case \n" );

 for ( i = 0; s[ i ] != '\0'; ++i )

 {

 printf( "%c", toupper( s[ i ]) );

 }

 printf (" \n\a end character codes of array s[] are: %d, %d \n\t ", s[i-2], s[i-1]);

 printf (" \n\a end characters of array s[] are: %c, %c \n\t", s[i-2], s[i-1]);

 printf (" \n\a decimal and hex address of end of 's' array: %u, %p \n\t ", &s[i], &s[i]);

 puts( "\n\a convert to lower case in reverse order" );

 j = i;

 for (j= i; j <-1; --j)

 {

 printf( "%c", tolower( s[ j ] ) );

 }

 puts("\t\a\n");

 printf( "\n \a\t The final value of loop variables 'i' and j are :%d\t%d\t:", i, j);

 puts( "\a\n");

 return 0;

 } // end main





**EEL 2880: Software Engineering: Case Study of Hardware and Software Integration**

**(Courtesy: Dr. Malcom Heimer: Design Architect of Pacemaker) Subbarao Wunnava 10 09 2019**

**Heart Scenario for Pacemaker Hardware/Software development**

**INTRODUCTION:**

Most bio-engineering and life-saving systems such as pace-makers, breathing controllers, and several such smart systems, operate with proper electronic sensors and efficient software and hardware integrations. The software should be very efficient and fault free; and minimal drain on the battery power. C language is preferred, as it is close to the machine code. These bio systems would have their own operating system, processor, memory and memory controllers, and operate with efficient machine code. The memory references are all in binary, referred by the hex-decimal coding.

**GENERAL CONSIDRATIONS**

 When heart beat and pumping activity changes [going low or high], electronic system called Pace Maker is implanted to monitor the heart condition and make appropriate correction for the heat condition. [Slow the heart rate, if it is running too fast, or speed up if it running too slow]. Pacemaker runs on battery with monitoring software, and works continuously to make the correction for the heart.

The code resident in the pacemaker is written in languages such as C, and assembled to be efficient machine code. Also, the program resides in Read Only Memory [ROM] of the pacemaker. The variables such as the heart rate, temperature, pumping condition etc are monitored by the software as I/O [Input/Output] and reside in pacemaker system RAM.

In USA and Canada, and rest of the world, pacemakers have changed the life styles, and longevity of heart patients. Pacemaker is a classic example of integration of hardware electronics, clean materials like electrodes and leads, efficient software optimizing the battery power, and fault tolerant software/hardware system.

